FM3/0+

Fan Motor Control

32-BIT MICROCONTROLLER
FM3/0+ Family

USER MANUAL

Publication Number S6E1A1_AN710-00004 Revision 1.4 Issue Date Aug 3, 2015

Target products

This application note describes the following products:

Series	Product Number (Not Including Package Suffix)
FM3 Series	MB9AF132K
FM0+ Series	S6E1A1

Table of Contents

Figures

Tables

1. Introduction

1.1 Purpose

This manual describes document architecture, demo connection and motor debug of Spansion DC Indoor Fan Motor solution. It can help you to understand and use Spansion Cortex M3/M0+ series MCU better. It can also provide some references to customers for development of new products.

1.2 Definitions, Acronyms and Abbreviations

API	Application Programming Interface
FOC	Field Oriented Control
HW	Hardware
FW	Firmware
I/O	Input and output
RAM	Random Access Memory

1.3 Document Overview

The rest of document is organized as the following:

Section 2 explains *System Environment*.

Section 3 explains *System Firmware Design.*

Section 4 explains *Event Function.*

Section 5 explains *User Interface.*

Section 6 explains *Interrupt Function.*

Section 7 explains *System Debug Sample*

Section 8 explains *System Error Code .*

2. System Environment

2.1 Hardware Environment

Table 2-1: MCU Hardware Development Environment

System	Description
MCU	S6E1A1
Hardware	FAN _Motor_SOLU-V0.2.1
Emulator	J-Link V8
Motor type	Fan motor 2#(RS:80Ω Ke:120; Ls:620)

2.2 Software Environment

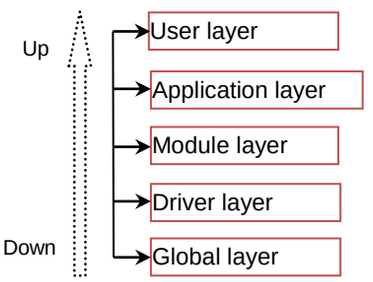
Table 2-2: MCU Software Development Environment

Type	Description
PC System	Windows 7 (64-bit)
Software developing IDE	IAR EWARM V6.6

3. System Firmware Design

3.1 Firmware Architecture

Figure 3-1: Relationship of the Files in Each Layer

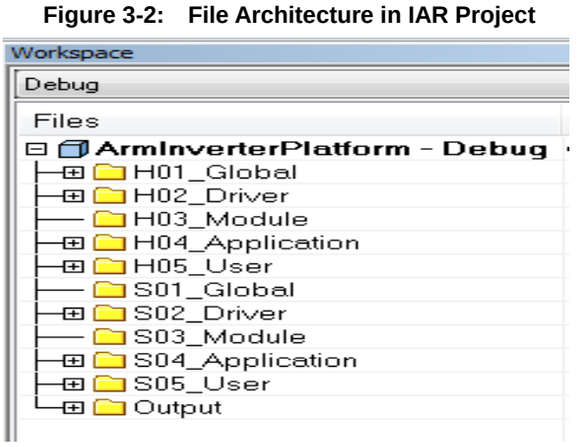


Each layer is independent. If the performance and independence will not be affected, the files in the lower layer can call the files in the upper layer. In most cases, the files in upper layer call the files in the lower layer.

3.1.1 IAR Project File Architecture

According to the firmware architecture model, the file in IAR project is composed of header file (.h), source file (.c) and the output file in build (.out).

Figure 3-2: File Architecture in IAR Project



Folder names are shown in Figure 3-2. The project includes five header folders and five source folders. The file in each folder is shown in the following table.

Table 3-1: Project File Architecture

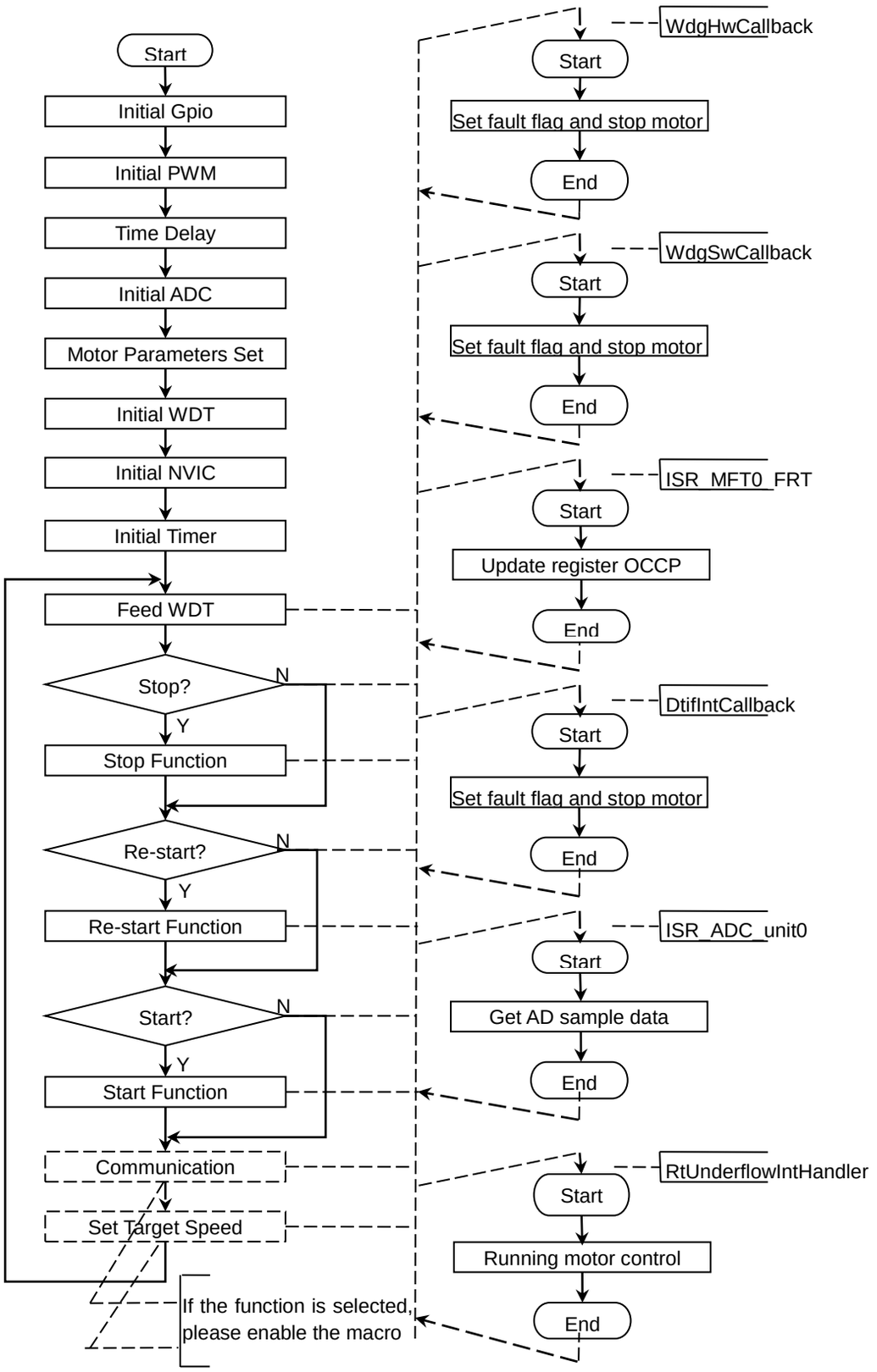
Folder Name	Description
H01-Global	Global function declaration, global variables and macro definition folder
H02-Driver	MCU function declaration, variables and macro definition folder
H03-Module	Event function declaration and variables definition folder
H04-Application	Application function declaration and variables definition folder
H05-User	Hardware parameters and software parameters and MCU configuration header file folder
S01-Global	Global function folder
S02-Driver	MCU driver function folder
S03-Module	Event function folder
S04-Application	Application function folder
S05-User	Hardware parameters and software parameters and MCU configuration source file folder

Remarks: The file Chip_Init.c in S05-User is MCU configuration file; ISR.c is interrupt entrance function file; Vector_Table.c is interrupt entrance vector table file, Customer_interface.c is motor control parameters and configuration file.

3.2 Firmware Flow Chart

The flow chart of main functions is shown in Figure 3-3.

Figure 3-3: The Main Loop Flow Chart



4. Event Function

4.1 Function List

Table 4-1: Function List Table

Prototype	Description	Remark
Clark Transform	Clarke Transform	N/A
Park Transform	Park Transform	N/A
InverseClackeTransform	Inverse Clarke transform	N/A
InverseParkTransform	Inverse Park transform	N/A
Sensor_Parameter_Initial	Current sensor parameters initialization	N/A
Motor1_Sensor_Offeset_Detect	Current sensor offset detect	N/A
Motor1_Sense	Current sensor function	N/A
DT_Compensation	Dead time compensation algorithm function	N/A
Dynamic_change_PWM	Dynamic change PWM carrier frequency function	N/A
Motor1_Elec_Brake	Motor electronic brake function	N/A
Field Weaken	Field weaken algorithm function	N/A
LogicRightShift	Bit logic right shift function	N/A
Sqrt	Square root function	N/A
Lpf	One order low pass filter	N/A
DifferParamsLpf	Differ parameters low pass filter	N/A
Initial_Motor1_FieldWeakPar	Field weaken function initialization	N/A
Initial_Motor1_BEMF_EstimPar	Position Estimator function initialization	N/A
Initial_Motor1_RunPar	Motor run function initialization	N/A
Motor1_Init	Motor total variables initialization function	N/A
Motor1_Start	Motor start up function	N/A
Motor1_Stop	Motor stop function	N/A
Motor1_Speed_Regulate	Motor speed regulate function	N/A
Motor1_Limit_Calc	Motor limit voltage and current function	N/A
Motor1_Theta_Generate	Motor theta generate function	N/A
Motor1_Process	Motor control main loop algorithm function	N/A
PositionObserver	Position Estimator event function	N/A
Motor1_HandleError	Motor error handler function	N/A
Motor1_Protect	Motor protect event function	N/A
Motor1_Run	Motor run and start up event function	N/A
NoStop_Start	No stop to restart function	N/A
Current_PI	Current loop PI controller function	N/A
Speed_PI	Speed loop PI controller function	N/A
u32Q22_ArcTan	Arctan function lookup algorithm	N/A
i32Q12_Sin	Sine function lookup algorithm	N/A
i32Q12_Cos	Cosine function lookup algorithm	N/A
Motor1_Res_Measure	Motor resistance measure function	N/A
Single_Shunt	Single shunt algorithm	N/A
Comm_Duration_Calc	Vector time calculate function	N/A
Sector_Judge	Sector judge algorithm function	N/A
abcon_map	Effective vector distribution function	N/A
SVPWM_Calc	SVPWM algorithm event function	N/A
Write_Motor1_MFT_Register	MFT register value assignment	N/A
InitNVIC	NVIC function mode initialization	N/A
InitGpio	Initialization I/O port	N/A
InitADC	Initialization AD model	N/A

InitSysTime	Clock mode initialization	N/A
InitWDT	Watch dog mode initialization	N/A
Motor1_configPWM	PWM carrier frequency configuration	N/A
Motor1_SVPWM_Init	Motor function module initialization	N/A
Motor1_SVPWM_En	Enable SVPWM mode	N/A
Motor1_SVPWM_Dis	Disable SVPWM mode	N/A
Speed_Feedback	Pulse feedback motor speed	N/A
Thermal_Protect	Thermal protection function	N/A
RMS_Calc	Root mean square calculate	N/A
Motor1_StartSpeed_PI	Motor start speed PI parameter adjustment	N/A
CAP_Charge	Bootstrap capacitance prepare charge	N/A
Main	Main loop function	N/A

4.2 Function Prototype

4.2.1 void ClarkeTransform()

Prototype	void ClarkeTransform(UVWAxiahCS *Input, AlphaBetaCS *Output);
Parameter	U-V-W stator coordinate variables, α-β stator coordinate variables
Return	N/A
Description	Clarke transform
Remark	N/A

4.2.2 void ParkTransform()

Prototype	void ParkTransform(AlphaBetaCS *Input, dqAxialCS *Output);
Parameter	α-β stator coordinate variables, d-q rotating coordinate variables
Return	N/A
Description	Park transform
Remark	N/A

4.2.3 void InvertClackeTransform()

Prototype	void InvertClackeTransform(AlphaBetaS *Input, UVWAxiahS *Output)
Parameter	α-β stator coordinate variables, u-v-w stator coordinate variables
Return	N/A
Description	Clarke inverse transform
Remark	N/A

4.2.4 void InvertParkTransform()

Prototype	void InvertParkTransform(dqAxialS *Input, AlphaBetaS *Output)
Parameter	d-q rotating coordinate variables, α-β stator coordinate variables
Return	N/A
Description	PARK inverse transform
Remark	N/A

4.2.5 void Sensor_Parameter_Initial ()

Prototype	void Sensor_Parameter_Initial (void)
Parameter	void
Return	void
Description	Current sampling parameters initialization
Remark	N/A

4.2.6 void Motor1_Sensor_Offeset_Detect()

Prototype	void Motor1_Sensor_Offeset_Detect(void)
Parameter	void
Return	void
Description	Current sampling offset detect
Remark	N/A

4.2.7 void Motor1_Sense()

Prototype	void Motor1_Sense(void)
Parameter	void
Return	void
Description	Current sampling event function
Remark	N/A

4.2.8 void DT_Compensation()

Prototype	void DT_Compensation(void)
Parameter	void
Return	void
Description	Dead time compensation algorithm function
Remark	N/A

4.2.9 void Dynamic_change_PWM()

Prototype	void Dynamic_change_PWM(void)
Parameter	void
Return	void
Description	Dynamic change PWM carrier frequency function
Remark	N/A

4.2.10 void Motor1_Elec_Brake ()

Prototype	void Motor1_Elec_Brake (void)
Parameter	void
Return	void
Description	Motor electronic brake function
Remark	N/A

4.2.11 void FieldWeaken()

Prototype	void FieldWeaken(FieldWeakenS * FieldWeakenPar, signed long Q8_Vbus, AlphaBetaS *Motor_2sVoltage, signed long *Motor_idref, unsigned char MotorRunningStage)
Parameter	Field weaken parameters
Return	void
Description	Field weaken algorithm function
Remark	N/A

4.2.12 int32_t LogicRightShift()

Prototype	int32_t LogicRightShift(int32_t Qmdata, uint32_t Shiftbits)
Parameter	Data input/Data output, data shift digits

Return	void
Description	Bit logic right shift function
Remark	N/A

4.2.13 unsigned long Sqrt ()

Prototype	unsigned long Sqrt (unsigned long Root)
Parameter	Data input/Data output
Return	void
Description	Square root function
Remark	N/A

4.2.14 void Lpf ()

Prototype	void Lpf(int32_t BeforeLpfData, int32_t *AfterLpfData, stc_lpfparams_t *pstcLpfParams)
Parameter	Input value, output value
Return	void
Description	One order low pass filter
Remark	N/A

4.2.15 void DifferParamsLpf ()

Prototype	void DifferParamsLpf(int32_t BeforeLpfData, int32_t *AfterLpfData, stc_differlpfparams_t *pstcLpfParams)
Parameter	Input value, output value, filter parameters
Return	void
Description	Differ parameters low pass filter
Remark	N/A

4.2.16 void Initial_Motor1_FieldWeakPar ()

Prototype	static void Initial_Motor1_FieldWeakPar(uint32_t sample_freq)
Parameter	Carrier frequency
Return	Void
Description	Field weaken function initialization
Remark	N/A

4.2.17 void Initial_Motor1_BEMF_EstimPar ()

Prototype	static void Initial_Motor1_BEMF_EstimPar(uint32_t sample_freq)
Parameter	Carrier frequency
Return	Void
Description	Position Estimator function initialization
Remark	N/A

4.2.18 void Initial_Motor1_RunPar ()

Prototype	static void Initial_Motor1_RunPar(uint32_t sample_freq)
Parameter	Carrier frequency
Return	Void
Description	Motor run function initialization
Remark	N/A

4.2.19 void Motor1_Init()

Prototype	void Motor1_Init(uint32_t Sample_freq)
Parameter	Carrier frequency
Return	Void
Description	Motor total variables initialization function
Remark	N/A

4.2.20 void Motor1_Start()

Prototype	void Motor1_Start(uint32_t Sample_freq)
Parameter	Carrier frequency
Return	void
Description	Motor start up function
Remark	N/A

4.2.21 void Motor1_Stop()

Prototype	void Motor1_Stop(void)
Parameter	Void
Return	void
Description	Motor stop function
Remark	N/A

4.2.22 void Motor1_Speed_Regulate ()

Prototype	void Motor1_Speed_Regulate (void)
Parameter	Void
Return	void
Description	Motor speed regulate function
Remark	N/A

4.2.23 void Motor1_Limit_Calc ()

Prototype	void Motor1_Limit_Calc (void)
Parameter	Void
Return	void
Description	Motor limit voltage and current function
Remark	N/A

4.2.24 void Motor1_Theta_Generate ()

Prototype	void Motor1_Theta_Generate (void)
Parameter	Void
Return	void
Description	Motor theta generate function
Remark	N/A

4.2.25 void Motor1_Process()

Prototype	void Motor1_Process(void)
Parameter	Void

Return	void
Description	Motor control main loop algorithm function
Remark	N/A

4.2.26 void PositionObserver ()

Prototype	void PositionObserver(stc_observer_t *pstcEstParams, AlphaBetaS *pstc2sV, AlphaBetaCS *pstc2sC)
Parameter	Position estimator parameters
Return	Void
Description	Position Estimator event function
Remark	N/A

4.2.27 void Motor1_HandleError ()

Prototype	void Motor1_HandleError(stc_Protect_t *pstcProtectParams)
Parameter	Motor protection parameters
Return	Void
Description	Motor error status handler function
Remark	N/A

4.2.28 void Motor1_Protect()

Prototype	void Motor1_Protect(stc_Protect_t *pstcProtectParams)
Parameter	Motor protection parameters
Return	void
Description	Motor protect event function
Remark	N/A

4.2.29 void Motor1_Run ()

Prototype	void Motor1_Run (void)
Parameter	Void
Return	void
Description	Motor run and start up event function
Remark	N/A

4.2.30 void NoStop_Start()

Prototype	void NoStop_Start(void)
Parameter	void
Return	void
Description	No stop to restart function
Remark	N/A

4.2.31 void Current_PI ()

Prototype	void Current_PI(int32_t Ref, int32_t Input, int32_t *Output, stc_pid_t *PID_Par)
Parameter	Reference current, feedback current, PI output, PI parameters
Return	void
Description	current loop PI controller function

Remark	N/A

4.2.32 void Speed_PI ()

Prototype	void Speed_PI(int32_t Ref, int32_t Input, int32_t *Output, stc_pid_t *PID_Par)
Parameter	Target speed, feedback speed, target torque
Return	void
Description	Speed loop PI controller function
Remark	N/A

4.2.33 uint32_t u32Q22_ArcTan()

Prototype	uint32_t u32Q22_ArcTan(int32_t i32Q8_Y,int32_t i32Q8_X)
Parameter	BEMF input, angle output
Return	void
Description	Arctan function lookup algorithm
Remark	N/A

4.2.34 int32_t i32Q12_Sin()

Prototype	int32_t i32Q12_Sin(uint32_t u32Q22_Angle)
Parameter	Angle input, angle sine value output
Return	void
Description	Sine function lookup algorithm
Remark	N/A

4.2.35 int32_t i32Q12_Cos ()

Prototype	int32_t i32Q12_Cos(uint32_t u32Q22_Angle)
Parameter	Angle input, angle cosine value output
Return	void
Description	Cosine function lookup algorithm
Remark	N/A

4.2.36 void Motor1_Res_Measure()

Prototype	void Motor1_Res_Measure(void)
Parameter	void
Return	void
Description	Motor resistance measure function
Remark	N/A

4.2.37 void Single_Shunt()

Prototype	void Single_Shunt(void)
Parameter	void
Return	void
Description	Single shunt algorithm
Remark	N/A

4.2.38 void Comm_Duration_Calc()

Prototype	void Comm_Duration_Calc(SVPWM_CalculateS *Motor_SVPWMPar)

Parameter	SVPWM parameters
Return	void
Description	Vector time calculate function
Remark	N/A

4.2.39 void Sector_Judge()

Prototype	void Sector_Judge(SVPWM_CalculateS *Motor_SVPWMPar, UVWAxiahS *Motor_3sVoltage)
Parameter	Three-phase voltage value
Return	void
Description	Sector judge algorithm function
Remark	N/A

4.2.40 void abcon_map()

Prototype	static inline void abcon_map(SVPWM_CalculateS *SVPWMPar)
Parameter	Vector value
Return	void
Description	Effective vector distribution function
Remark	N/A

4.2.41 void SVPWM_Calc()

Prototype	Void SVPWM_Calc(SVPWM_CalculateS *SVPWMPar ,AlphaBetaS *_2sV, UVWAxiahS *_3sV, AlphaBetaS *_2sV_Real)
Parameter	SVPWM parameters
Return	void
Description	SVPWM algorithm event function
Remark	N/A

4.2.42 void Write_Motor1_MFT_Register()

Prototype	void Write_Motor1_MFT_Register(SVPWM_CalculateS *SVPWM_Par)
Parameter	Effective Vector value
Return	void
Description	MFT register value assignment
Remark	N/A

4.2.43 void InitNVIC ()

Prototype	void InitNVIC (void)
Parameter	void
Return	void
Description	NVIC function mode initialization
Remark	N/A

4.2.44 void InitGpio ()

Prototype	void InitGpio(void)
Parameter	void

Return	void
Description	Initialization I/O port
Remark	N/A

4.2.45 void InitADC ()

Prototype	void InitADC(uint32_t Motor1_Sample_freq)
Parameter	PWM carrier frequency
Return	void
Description	ADC module initialization
Remark	N/A

4.2.46 void InitSysTime ()

Prototype	void InitSysTime (void)
Parameter	void
Return	void
Description	System timer initialization
Remark	N/A

4.2.47 void InitWDT ()

Prototype	void InitWDT (void)
Parameter	void
Return	void
Description	Watch dog mode initialization
Remark	N/A

4.2.48 void Motor1_configPWM ()

Prototype	void Motor1_configPWM(uint32_t parameter)
Parameter	Current loop frequency
Return	void
Description	PWM carrier frequency configuration
Remark	N/A

4.2.49 void Motor1_SVPWM_Init ()

Prototype	void Motor1_SVPWM_Init (void)
Parameter	void
Return	void
Description	Motor function module initialization
Remark	N/A

4.2.50 void Motor1_SVPWM_En ()

Prototype	void Motor1_SVPWM_En (void)
Parameter	void
Return	void
Description	Enable SVPWM mode
Remark	N/A

4.2.51 void Motor1_SVPWM_Dis ()

Prototype	void Motor1_SVPWM_Dis (void)
Parameter	void
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.52 void Speed_Feedback()

Prototype	void Speed_Feedback(stc_speedfeedback_t *pstcSpeedParams)
Parameter	Pulse count value
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.53 void Thermal_Protect()

Prototype	void Thermal_Protect(stc_thermalprotection_t *pstcThermalParams)
Parameter	Thermal sensor parameters
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.54 void RMS_Calc()

Prototype	void RMS_Calc(int32_t angle, int32_t data,stc_rmsparams_t *rms)
Parameter	Motor rotor angle, motor one phase current, RMS function parameters.
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.55 void Motor1_StartSpeed_PI()

Prototype	void Motor1_StartSpeed_PI(stc_parameters_t *startspeedpi)
Parameter	Start speed PI adjustment function parameters
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.1 void CAP_Charge ()

Prototype	void CAP_Charge(stc_capparams_t *charge)
Parameter	CAP charge function parameters
Return	void
Description	Disable SVPWM mode
Remark	N/A

4.2.2 void Main ()

Prototype	void Main(void)
Parameter	void
Return	void
Description	Main loop function
Remark	N/A

5. User Interface

5.1 Customer Interface Variable List

Motor1: (Fan)

Motor Parameters

Motor1_Pole_Pairs	- Motor pole pairs
Motor1_Ld	- PMSM d axis phase inductance
Motor1_Lq	- PMSM q axis phase inductance
Motor1_Rs	-PMSM line resistor
Motor1_Ke	- PMSM Back Electromotive Force coefficient
Motor1_Kt	- PMSM torque const
Motor1_Kj	- PMSM moment of inertia

PI Parameters Regulate information

Motor1_dki	- D axis current loop integral parameter
Motor1_dkp	- D axis current loop proportion parameter
Motor1_qki	- Q axis current loop integral parameter
Motor1_qkp	- Q axis current loop proportion parameter
Motor1_ski	- Speed loop integral parameter
Motor1_skp	- Speed loop proportion parameter
Motor1_Speed_Inc_Acceleration_Hz	- Acceleration of speed loop
Motor1_Speed_Dec_Acceleration_Hz	- Deceleration of speed loop

Motor Startup Parameters

Motor1_OpenLoop_Inc_Current_APS	- Current vary step in open loop
Motor1_Openloop_CurrentA	- Current value in open loop
Motor1_Orient_TimeS	- Orient time
Motor1_Force_Inc_Speed_Hz	- Open loop acceleration
Motor1_CoseLoop_Speed_Hz	- Target speed when switching to closed loop
Motor1_Initial_Speed_Pre_Close_Hz	- Speed initial value before closed loop
Motor1_Detect_Direction_TimeS	- Detection time of rotational direction
Motor1_Inverse_Rotor_Brake_TimeS	- Brake time of motor reversion
Motor1_Pre_CloseLoop_TimeS	- Time before closed loop
Motor1_Pre_CloseLoop_Current_MinA	- Minimum current before closed loop
Motor1_Pre_CloseLoop_Current_MaxA	- Maximum current before closed loop
Motor1_Inc_SpeedAcc_TimeS	- Acceleration interval
Motor1_Inc_Speed_MAX_Hz	- Maximum speed acceleration
Motor1_PWM_Brake_Inc_TimeS	- PWM brake mode brake duty ratio pulse increase time
Motor1_Stop_Brake_CurrentA	- Motor braking stops when the motor braking current is low

Motor1 protection parameters

Motor1_Over_CurrentA	- Software over current point
Motor1_max_dcvoltageV	- DC bus over voltage point
Motor1_min_dcvoltageV	- DC bus under voltage point
Motor1_Error_keep_timeS	- Fault maintain time

Motor1 running parameters

Motor1_Field_Value	- Maximum current of field value
Motor1_Comp_MaxA	- Dead time compensation offset balance current
Motor1_Is_MaxA	- Motor running maximum torque current
Motor1_dead_timeUS	- Dead time value
Motor1_PWM_Carry_Frequency	- Motor start running PWM carry frequency
Motor1_Current_Carry_Frequency	- Current implement frequency
Motor1_speed_Max_RPM	-Motor running maximum speed
Motor1_speed_Min_RPM	-Motor running minimum speed
Motor1_Rotor_direction	-Motor running direction set

Motor1_Running_Level	-Set motor running stage
Motor1_Dynamic_PWM_Enable	-Motor carrier dynamic change switching
Motor1_PWM_Table	-Motor carrier frequency table

5.2 Variable Definition

5.2.1 Motor1_Pole_Pairs

Data type: unsigned char

Description: PMSM pole pairs

Unit: none

Measure method: Turn motor rotor manually, test motor three-phase terminal line voltage waveform, record turning revolution N and sine wave period number M, so

Motor1_pole_pairs = M/N

5.2.2 Motor1_Ld

Data type: float

Description: PMSM d axis phase inductance

Unit: mH

Note: This parameter can be obtained from motor parameters or testing manually.

5.2.3 Motor1_Lq

Data type: float

Description: PMSM q axis phase inductance

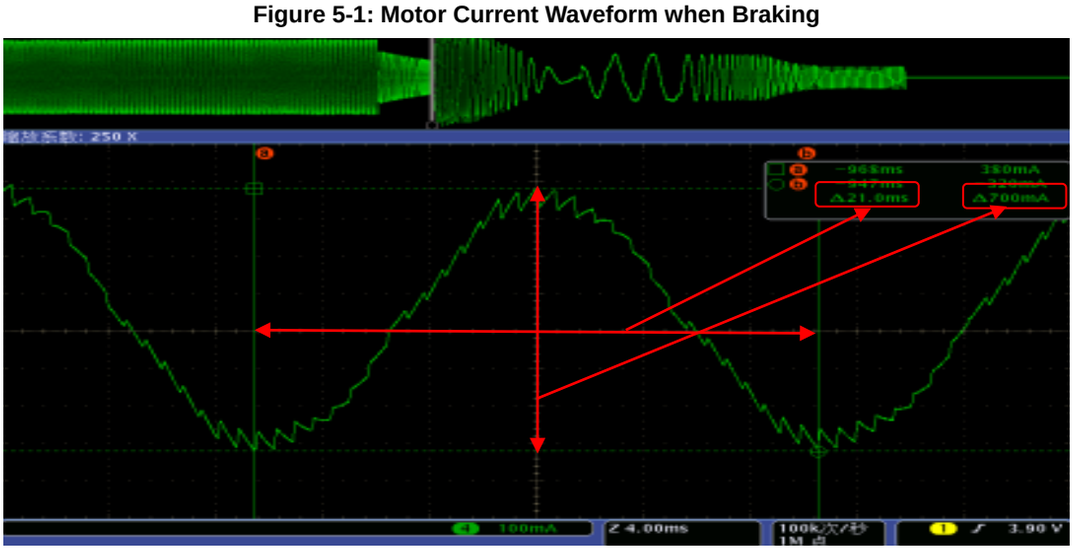
Unit: mH

Note: This parameter can be obtained from motor parameters or test results.

Measure method:

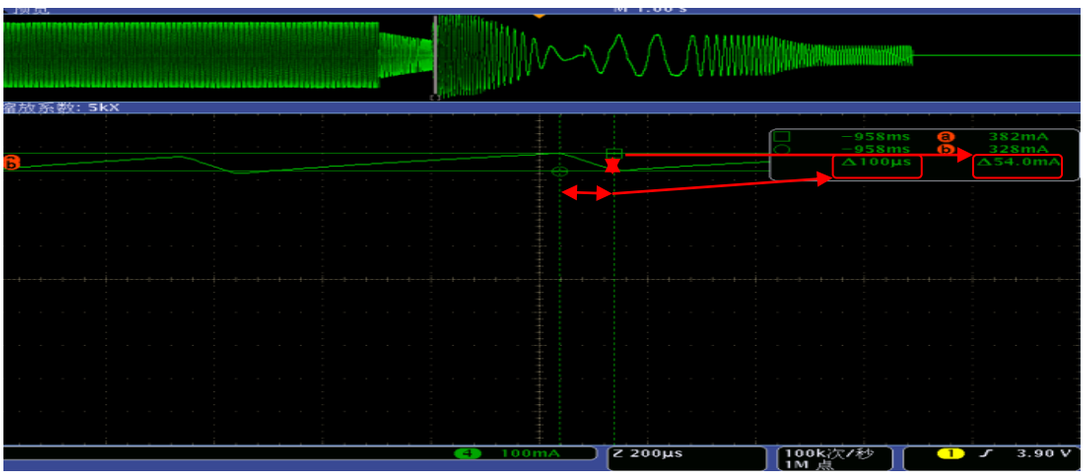
When motor is running, brake motor manually and measure the phase current waveform. The current waveform is shown in Figure 5-1. Then Lq is calculated by using braking phase current. This method is only applicable to SPM motor.

Figure 5-1: Motor Current Waveform when Braking



When motor braking, the phase current waveform detail is shown in Figure 5-2

Figure 5-2: Motor Current Waveform Detail



Considering Ke, Rs and Motor1_pole_pairs are known, record the following data when motor braking:

The frequency of current waveform: F

The current falling time: T

The current amplitude changes in falling time: A

The peak-peak current: A0

So Lq is calculated as follows,

Motor1_Lq = (F*60/ Motor1_pole_pairs* Motor1_Ke/1000 - Motor1_Res* A0/2)*T/A

5.2.4 Motor1_Res

Data type: float

Description: PMSM line resistor

Unit: Ω

Measure method: Test any two phases of resistance using multi-meter, record three test results Ruv, Rvw, Ruw, so

Motor1_Res = (Ruv +Rvw+ Ruw)/3

5.2.5 Motor1_Ke

Data type: float

Description: PMSM Back Electromotive Force coefficient

Unit: V/Krpm

Measure method: Turn motor rotor rapidly, test motor three-phase terminal line voltage waveform, and choose a relatively stable range of line voltage amplitude, record voltage peak-peak value and frequency in the range.

Motor1_Ke = V* Motor1_pole_pairs*5.8934465/F

5.2.6 Motor1_Kt

Data type: float

Description: PMSM torque constant

Unit: N·m/Arms

5.2.7 Motor1_Kj

Data type: float

Description: PMSM moment of inertia

Unit: Kg·m2

5.2.8 Motor1_dki

Data type: float

Description: D axis current loop integral parameter

Unit: none

Range: about 0.3

5.2.9 Motor1_dkp

Data type: float

Description: D axis current loop proportion parameter

Unit: none

Range: about 0.3

5.2.10 Motor1_qki

Data type: float

Description: Q axis current loop integral parameter

Unit: none

Range: about 0.3

5.2.11 Motor1_qkp

Data type: float

Description: Q axis current loop proportion parameter

Unit: none

Range: about 0.3

5.2.12 Motor1_ski

Data type: float

Description: Speed loop integral parameter

Unit: none

Range: about 0.0005

5.2.13 Motor1_skp

Data type: float

Description: Speed loop proportion parameter

Unit: none

Range: about 0.3

5.2.14 Motor1_Speed_Inc_Acceleration_Hz

Data type: float

Description: Acceleration of speed loop

Unit: Hz

5.2.15 Motor1_Speed_Dec_Acceleration_Hz

Data type: float

Description: Deceleration of speed loop

Unit: Hz

5.2.16 Motor1_OpenLoop_Inc_Current_APS

Data type: float

Description: Current vary step in open loop

Unit: A

5.2.17 Motor1_OpenLoop_CurrentA

Data type: float

Description: Current value in open loop

Unit: A

5.2.18 Motor1_Orient_TimeS

Data type: float

Description: Orient time

Unit: s

5.2.19 Motor1_Force_Inc_Speed_Hz

Data type: float

Description: PMSM open loop acceleration. According to motor inertia, the greater inertia of the motor is, the smaller the parameter value becomes.

Unit: Hz

5.2.20 Motor1_CloseLoop_Speed_Hz

Data type: float

Description: Target speed when switching to closed loop. The parameter value is greater than or equal to the minimum frequency of the motor.

Unit: Hz

5.2.21 Motor1_Initial_Speed_Pre_Close_Hz

Data type: float

Description: PMSM target speed before close loop, if the value is less than minimum close loop speed, system uses the open loop reference current and current acceleration as the close loop premise. If the value is greater than or equal to the close loop target speed, system will start up with prepared close loop model.

Unit: Hz

5.2.22 Motor1_Detect_Direction_TimeS

Data type: float

Description: Detection time of rotational direction. If set the value to zero, system will close brake function by default.

Unit: s

5.2.23 Motor1_Inverse_Rotor_Brake_TimeS

Data type: float

Description: PMSM rotor break duration, which is based on the motor current and the speed in the breaking moment.

Unit: s

5.2.24 Motor1_Pre_CloseLoop_TimeS

Data type: float

Description: Time before closed loop

Unit: s

5.2.25 Motor1_Pre_CloseLoop_Current_MinA

Data type: float

Description: Minimum current before closed loop

Unit: A

5.2.26 Motor1_Pre_CloseLoop_Current_MaxA

Data type: float

Description: Maximum current before closed loop

Unit: A

5.2.27 Motor1_Inc_SpeedAcc_TimeS

Data type: float

Description: motor open loop speed increase interval

Unit: s

5.2.28 Motor1_Inc_Speed_MAX_Hz

Data type: float

Description: motor maximum speed acceleration

Unit: Hz

5.2.29 Motor1_PWM_Brake_Inc_TimeS

Data type: float

Description: PMSM PWM break duty acceleration, if the value is greater than 2S, system uses the zero vector break mode as default; if the value is less than 2S, system uses the PWM break mode. The less the value is, the greater the effect becomes.

Unit: S

5.2.30 Motor1_Stop_Brake_CurrentA

Data type: float

Description: Motor braking stops when the motor braking current is smaller than “Motor1_Stop_Brake_CurrentA”.

Unit: S

5.2.31 Motor1_Over_CurrentA

Data type: float

Description: Software over current point

Unit: A

5.2.32 Motor1_max_dcvoltageV

Data type: unsigned short

Description: DC bus over voltage point

Unit: V

5.2.33 Motor1_min_dcvoltageV

Data type: unsigned short

Description: DC bus under voltage point

Unit: V

5.2.34 Motor1_Error_keep_timeS

Data type: unsigned short

Description: DC bus under voltage point

Unit: V

5.2.35 Motor1_Field_Value

Data type: float

Description: Maximum current of field weaken

Unit: A

5.2.36 Motor1_Comp_MaxA

Data type: float

Description: the dead time compensation value of current

Unit: A

5.2.37 Motor1_Is_MaxA

Data type: float

Description: Motor maximum current

Unit: A

5.2.38 Motor1_dead_timeUS

Data type: float

Description: PWM dead time

Unit: us

5.2.39 Motor1_PWM_Carry_Frequency

Data type: unsigned short

Description: PWM carrier frequency

Unit: Hz

5.2.40 Motor1_Current_Carry_Frequency

Data type: unsigned short

Description: Motor control algorithm frequency

Unit: Hz

5.2.41 Motor1_speed_Max_RPM

Data type: unsigned short

Description: Maximum speed

Unit: RPM

5.2.42 Motor1_speed_Min_RPM

Data type: unsigned short

Description: Minimum speed

Unit: RPM

5.2.43 Motor1_Rotor_direction

Data type: unsigned char

Description: Motor rotate direction

Unit: none

Range: 0->clockwise, 1->anticlockwise

5.2.44 Motor1_Running_Level

Data type: unsigned char

Description: Motor running level

Unit: none

Range: 0->Brake, 1->Orient, 2->Open loop, 3->Closed loop, 4->Set running speed

5.2.45 Motor1_Dynamic_PWM_Enable

Data type: unsigned char

Description: switch of the Dynamic PWM function

Unit: none

Range: 0->disable, 1->enable

5.2.46 Motor1_PWM_Table

Data type: unsigned short

Description: the carry frequency of per frequency

Unit: KHZ

5.2.47 AD_OFFSET_NUM

Data type: macro definition

Description: Stationary phase current will be sampled 2 AD_OFFSET_NUM times, before the AD middle voltage is calculated.

Unit: None

5.2.48 AD_OFFEST_MAX_VALUE

Data type: macro definition

Description: AD middle voltage maximum offset

Unit: None

5.2.49 Motor1_Control_mode

Data type: macro definition

Description: Motor1 control mode switch

Range: 0: Speed 1: Torque 2: VF 3: VF_Diagnose

5.2.50 MOTOR1_PWM_POLAR_AVAILABLE

Data type: macro definition

Description: Motor1 control PWM output polarity

Range: 0: LOW_POLAR 1: HIGH_POLAR

5.2.51 SYS_CLOK

Data type: macro definition

Description: MCU main frequency

Unit: MHz

Range: 0—20MHz

5.2.52 MOTOR1_SHUNT_NUMBER

Data type: macro definition

Description: The number of sampling resistor

Unit: none

Range: 1, 2, 3

5.2.53 SPEED_FROM_OUT

Data type: macro definition

Description: modify the source of motor target speed

Unit: none

Range: 0: from the UART or debug set 1: from the out AD

5.2.54 VSP_MOTOR_START_V

Data type: macro definition

Description: The voltage threshold value of motor start in Vsp control speed model.

Unit: V

Range: 0-5V

5.2.55 PULSE_FEEDBACK_ENABLE

Data type: macro definition

Description: motor speed count uses pulse counting model or not, if the parameter is enabled, there are 12 PWM cycle feedbacks from pin ‘P61’ in one motor mechanical cycle.

Unit: none

Range: 0 – disable; 1 – enable.

5.2.56 MOTOR1_IDENTIFICATION_ENABLE

Data type: macro definition

Description: motor parameters identification function enable flag.

Unit: none

Range: 0 – disable; 1 – enable.

5.2.57 MOTOR1_THERMAL_PROTECTION

Data type: macro definition

Description: thermal protect function enable flag, if the parameters are enabled, when the temperature of the motor rises, the speed will slow down; if the temperature exceeds the max value, motor will stop running to protect itself.

Unit: none

Range: 0 – disable; 1 – enable.

5.2.58 MOTOR1_U_PIN

Data type: macro definition

Description: AD Channel to sample U phase current

Unit: MCU pin

Range: ANXX

5.2.59 MOTOR1_V_PIN

Data type: macro definition

Description: AD Channel to sample V phase current

Unit: MCU pin

Range: ANXX

5.2.60 MOTOR1_W_PIN

Data type: macro definition

Description: AD Channel to sample W phase current

Unit: MCU pin

Range: ANXX

5.2.61 DC_V_PIN

Data type: macro definition

Description: AD Channel to sample DC bus voltage

Unit: MCU pin

Range: ANXX

5.2.62 MOTOR1_SPEED_PIN

Data type: macro definition

Description: AD channel to sample sliding rheostat to control motor speed

Unit: MCU pin

Range: ANXX

5.2.63 ADC_Digit

Data type: macro definition

Description: AD converter resolution

Unit: bit

Range: 1-bit, 12-bit

5.2.64 ADC_REF

Data type: macro definition

Description: ADC reference voltage

Unit: V

Range: 1.8V/3.3V/5V

5.2.65 VDC_Amplifier_Multiple

Data type: macro definition

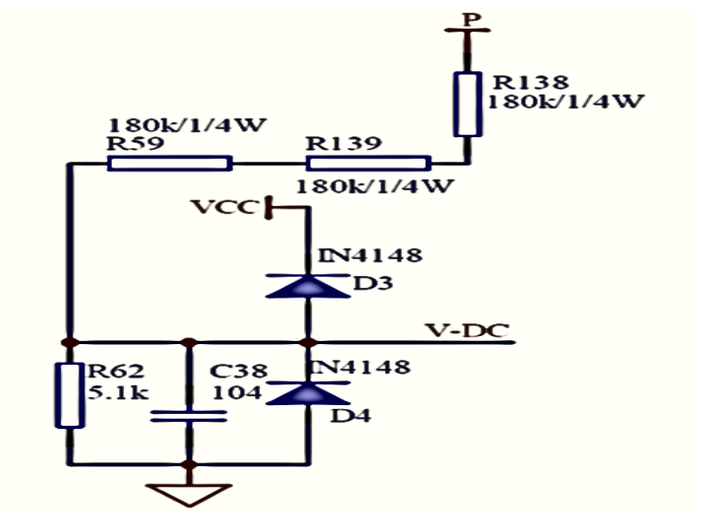
Description: DC bus voltage amplifying circuit coefficient

Unit: none

Calculation process: as shown in Figure 5-3;

VDC_Amplifier_Multiple = (R138+R39+R59+R62)/R62;

Figure 5-3: DC Bus Voltage Amplifying Circuit



5.2.66 Motor1_Current_Rs

Data type: macro definition

Description: Current sampling resistance of Motor1

Unit: Ω

Note: the shunt connect to IPM in circuit as shown in Figure 5-4; if use built-in shunt, please refer to IPM datasheet to get the shunt resistance.

Figure 5-4: Phase Current Sampling Circuit

5.2.67 Motor1_Current_Amplifier_Multiple

Data type: macro definition

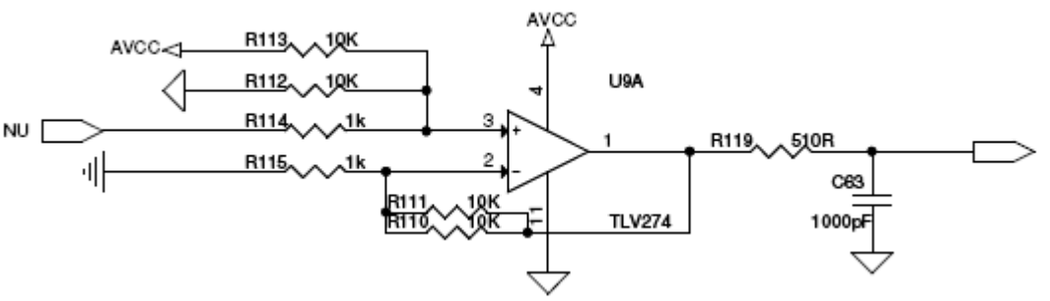
Description: Current amplifying circuit coefficient

Unit: none

Calculation process: as shown in Figure 5-5,

Motor1_Current_Amplifier_Multiple = (R111//R110)/R115;

Figure 5-5: Current Amplifying Circuit



5.2.68 Motor1_Pre_Charge_TimeS

Data type: macro definition

Description: IPM bootstrap capacitor charge time in motor start-up status.

Unit: S

Range: according to IPM characteristic and hardware parameters.

5.2.69 Motor1_Pre_Charge_Duty_Cycles

Data type: macro definition

Description: IPM bootstrap capacitor charge PWM duty

Unit: None

Range: 0~1

5.2.70 MOTOR1_BEMF_LPFK_MIN

Data type: macro definition

Description: the minimum of motor1 estimate filters K value

Unit: None

Range: according to motor characteristic, hardware and sensor circuit performance

5.2.71 MOTOR1_BEMF_LPFK_MAX

Data type: macro definition

Description: the maximum of motor1 estimate filters K value

Unit: None

Range: according to motor characteristic, hardware and sensor circuit performance

5.2.72 MOTOR1_BEMF_LPFK_MIN_HZ

Data type: macro definition

Description: The minimum value of rotor position estimator’s filter

Unit: None

Range: equal to motor lowest run frequency

5.2.73 MOTOR1_BEMF_LPFK_MAX_HZ

Data type: macro definition

Description: The maximum value of rotor position estimator’s filter

Unit: None

Range: equal to motor lowest run frequency

6. Interrupt Function

6.1 Function List

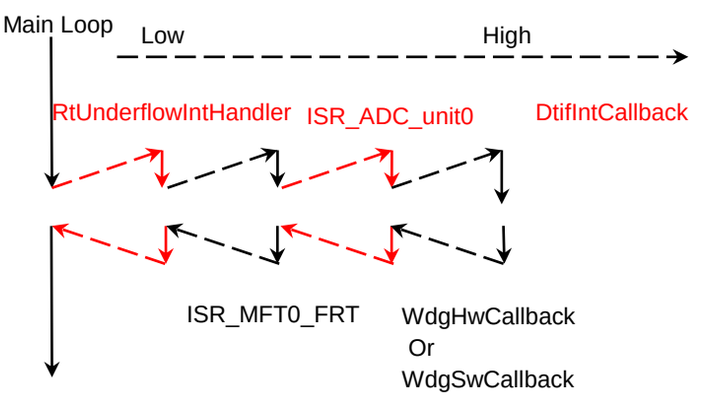
Table 6-1: Interrupt Function Table

Function Prototype	Description	Remark
WdgHwCallback	Hard watch dog interrupt	N/A
WdgSwCallback	Soft watch dog interrupt	N/A
ISR_MFT0_FRT	FRT interrupt	N/A
DtifIntCallback	DTTI interrupt	N/A
ISR_ADC_unit0	ADC0 interrupt	N/A
RtUnderflowIntHandler	Base timer interrupt	N/A

6.2 Interrupt Priority Configuration

1. Interrupt Priority Map

Figure 6-1: Interrupt Priority Map



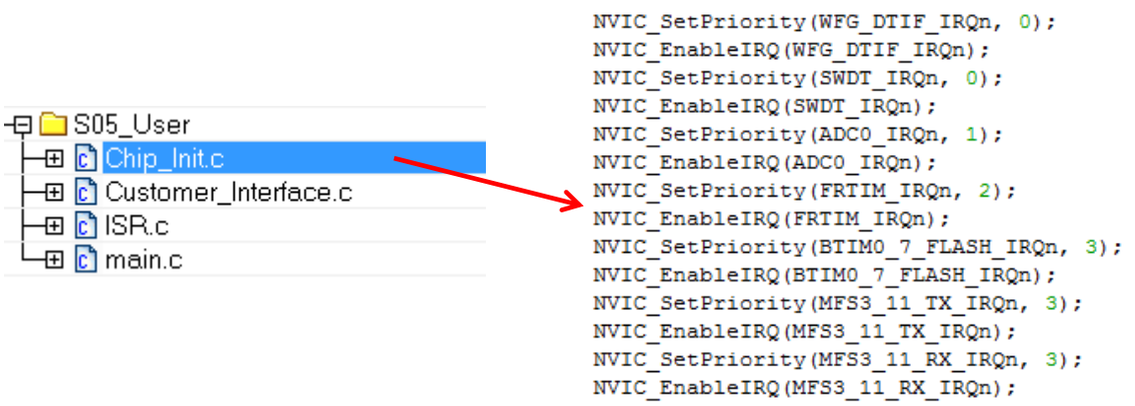
2. Interrupt Priority List

Table 6-2: Interrupt Priority Table

Interrupt Vector	Priority	
	M3	M0+
DtifIntCallback	0	0
WdgHwCallback Or WdgSwCallback	1	0
ISR_ADC_unit0	2	1
ISR_MFT0_FRT	3	2
RtUnderflowIntHandler	4	3

3. Interrupt Priority Configuration Location

Figure 6-2: Interrupt Priority Configuration Diagrammatic Sketch (FM0+)



7. System Debug Sample

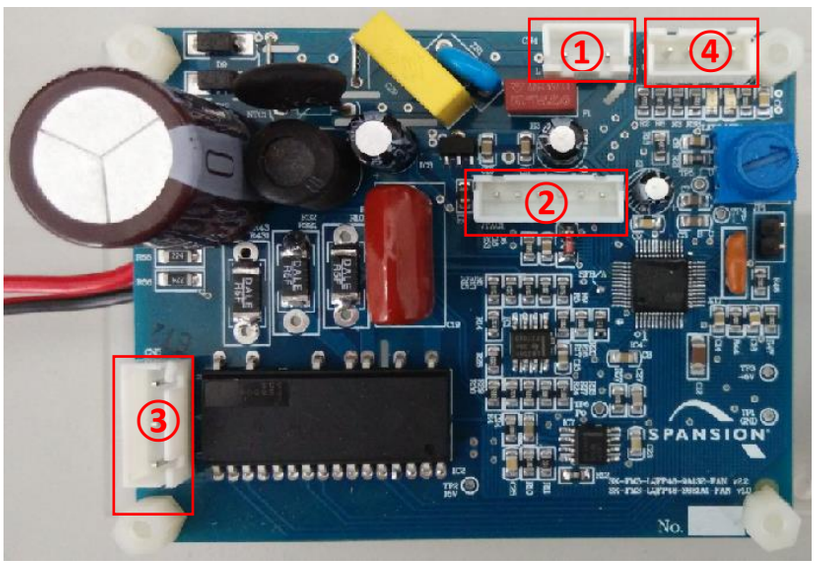
7.1 System Build

7.1.1 Demo Board Connect

When you get a Fan Motor board as shown in Figure 7-1, please connect as follows:

① Power supply interface;
② Emulator debug interface;
③ Motor three phase line interface;
④ GUI debug interface.

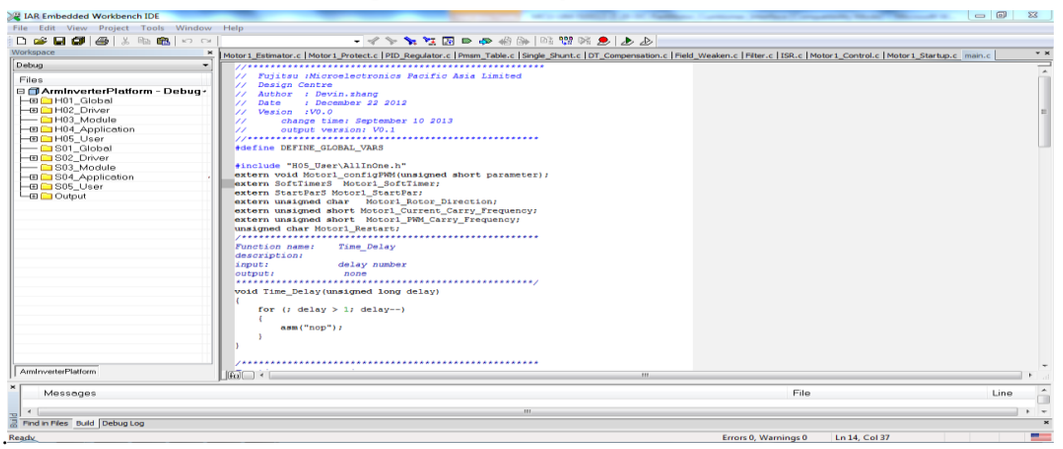
Figure 7-1: Front View of Demo Board



7.1.2 Open Project

Firstly, run IAR Embedded Workbench, click “File->Open->Workspace”, choice the project name “ArmInverterPlatform.eww”, the opened project interface is show in Figure 7-2.

Figure 7-2: IAR Project Interface

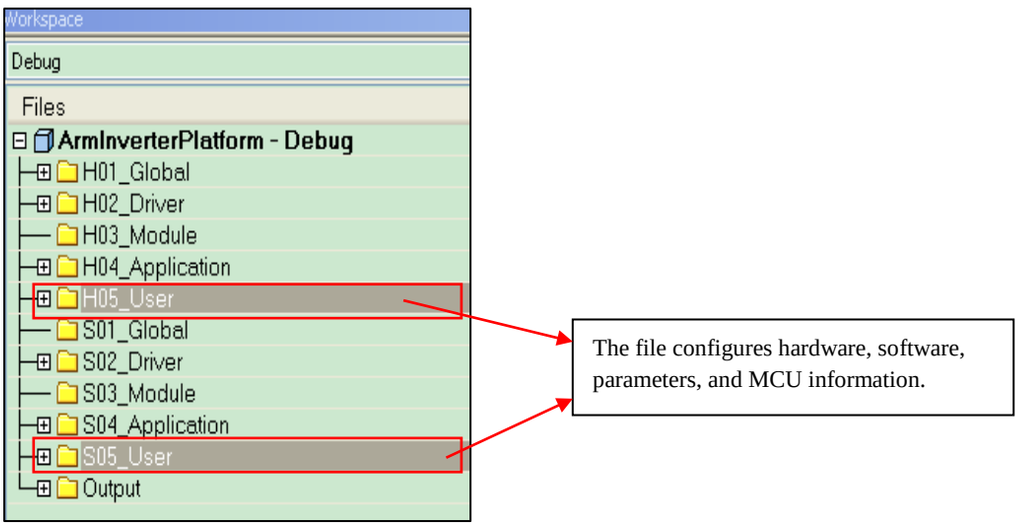


7.1.3 Configure Hardware and Software

1. Configure Motor Interface Location

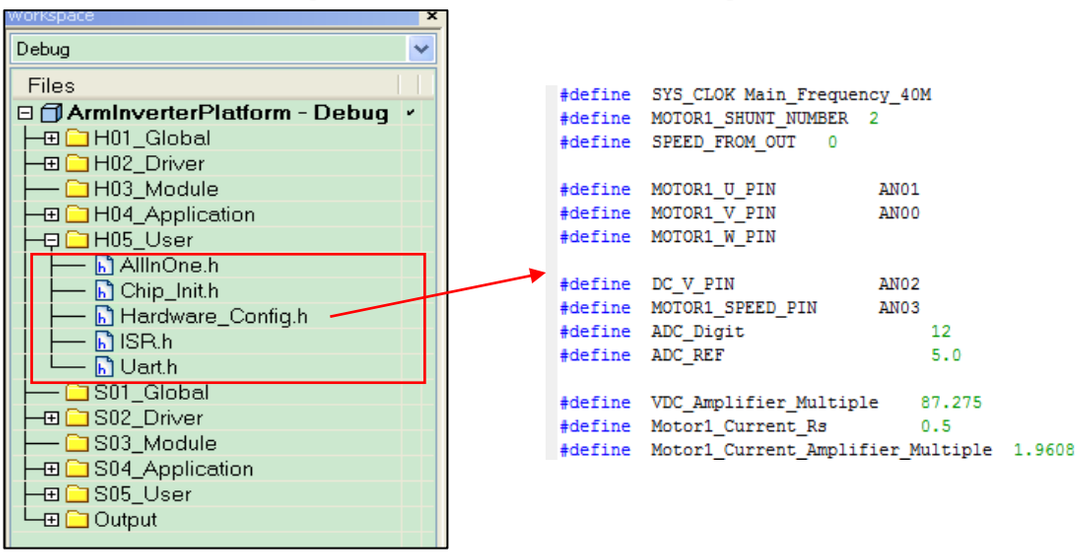
When you change the motor or hardware, please modify the related file as shown in Figure 7-3.

Figure 7-3: Project Structure



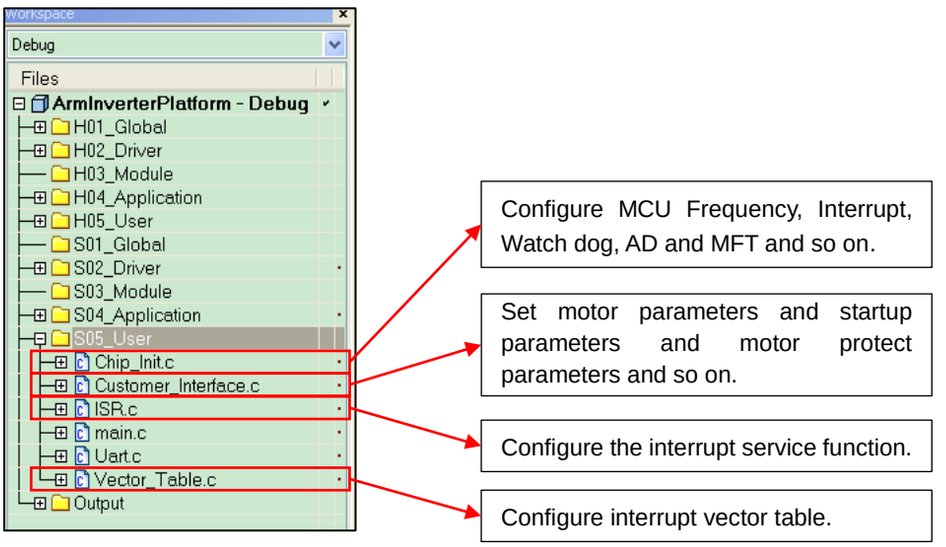
2. Configure hardware parameters as shown in Figure 7-4

Figure 7-4: Hardware Parameters Configuration



3. Configure MCU Register, Interrupt and motor parameters location. As shown in Figure 7-5;

Figure 7-5: Configure MCU Register, Interrupt and Motor Parameters File Structure



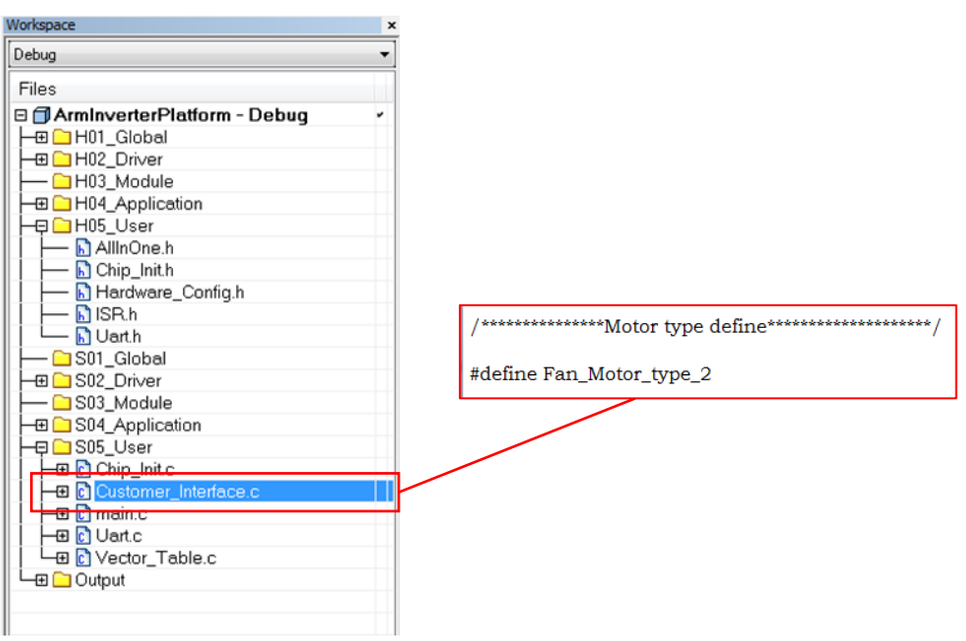
7.1.4 Configure User Interface

7.1.4.1 Configure Motor Type

1. Define the motor type in file “Customer_Interface.c”,

For example: define the motor name to “Fan_Motor_Type_2”, as shown in Figure 7-6;

Figure 7-6: Define Motor Name



2. Please copy and paste a data set as shown in Figure 7-7, modify the motor pole, reluctance, inductance and induction constant. In this project, we do not care about torque constant and inertia constant.

Motor parameter can be obtained from motor factory directly, or measured by the user.

Figure 7-7: Define Motor Parameters

```
/***************Motor Paramters********************/
uint32_t        Motor1_Pole_Pairs =  4.0;          //the pole pairs of rotor
float32_t       Motor1_Ld   = 107;                   //the d axis reductanc
float32_t       Motor1_Lq   = 107;                    //the q axis reductance
float32_t       Motor1_Rs   = 52;                   //the resistance between two phases(U,V eg.)
float32_t       Motor1_Ke   = 120;                   //inductive voltage constant
float32_t       Motor1_Kt   = 0.39;                 //torque constant
float32_t       Motor1_Kj   = 0.685;                 //moment of inertia constant
```

7.1.4.2 Configure PI Parameters

You do not need to modify the default PI parameters, when the motor is running, system can modify the PI parameters of speed loop and current loop automatically according to motor running state. The PI parameters of speed loop are based on motor inertia, and the PI parameters of current loop are based on the resistance and inductance. The detail information is shown in chapter 7.2.4.2.

Figure 7-8: Motor PI Parameters

```
/***************PI Paramters Regulate information********************/
float32_t       Motor1_Dki  = 6.0;                   //d axis current PI regulator integral con
float32_t       Motor1_Dkp  = 300.0;                 //d axis current PI regulator proportion c
float32_t       Motor1_Qki  = 6.0;                   //q axis current PI regulator integral con
float32_t       Motor1_Qkp  = 300.0;                 //q axis current PI regulator proportion c
float32_t       Motor1_Ski = 0.0005;                 //speed PI regulator integral constant
float32_t       Motor1_Skp = 0.1;                    //speed PI regulator proportion constant
float32_t       Motor1_Speed_Inc_Acceleration_Hz = 5.0;    //the speed increase acceleration frequency
float32_t       Motor1_Speed_Dec_Acceleration_Hz = 5.0;    //the speed decrease acceleration frequency
```

7.1.4.3 Configuration Start-up Parameters

1. Start-up with orient and open loop; set the orient duration and open loop current;

2. According to motor inertia and motor minimum operation frequency, set the acceleration and close loop frequency with open loop;

3. Start-up with anti-wind, define the detect direction times;

   a. Do not need Start-up with anti-wind, define it to zero;

   b. Need start-up with anti-wind function, define the detect direction times as a non-zero value. In principle, if the time can satisfy the direction detection, please set the time as short as possible, the typical case is 0.2S.

4. Start-up with close-loop, set Pre close loop time and prepare close loop current. For the detail of open loop acceleration and maximum speed with open loop and PWM break duty cycle, please refer to chapter 5.2.

Figure 7-9: Motor Start-up Parameters

```
/***************Motor Startup Parameter*********************/
float32_t  Motor1_OpenLoop_Inc_Current_APS = 5.2;       //the openloop stage current increase value
float32_t  Motor1_OpenLoop_CurrentA       = 0.5;       //the openloop stage maximum current
float32_t  Motor1_Orient_TimeS            = 0;          //the motor start up orient stage time of openloop stage
float32_t  Motor1_Force_Inc_Speed_Hz =  0.1;            //the increase speed of openloop stage
float32_t  Motor1_CloseLoop_Speed_Hz =  2.0;            //the open loop frequency change to close loop
float32_t  Motor1_Initial_Speed_Pre_Close_Hz =2;        //the open loop increase speed and pre close module control
float32_t  Motor1_Detect_Direction_TimeS =0.15;          //the rotor initial status detection time
float32_t  Motor1_Inverse_Rotor_Brake_TimeS =0.1;         //the brake maximum time
float32_t  Motor1_Pre_CloseLoop_TimeS = 2.0;            //the pre close stage maintain time
float32_t  Motor1_Pre_CloseLoop_Current_MinA =0;        //the close loop stage minimum current value
float32_t  Motor1_Pre_CloseLoop_Current_MaxA =0.5;     //the pre close stage maximum current value
float32_t  Motor1_Inc_SpeedAcc_TimeS =  0.0018;         //the speed accelerator increase time
float32_t  Motor1_Inc_Speed_MAX_Hz =    1.2;            //the speed accelerator maximum value
float32_t  Motor1_PWM_Brake_Inc_TimeS = 2;              //the brake pwm pulse increase time
float32_t  Motor1_Stop_Brake_CurrentA = 0.045;         //motor stop braking when motor brake current litter.
```

7.1.4.4 Configuration Protection Parameters

Configuration principles: according to the characteristics of motor, the threshold of current and voltage can't cause damage to motor.

Figure 7-10: Motor Protection Parameters

```
/**************motor1 protection parameters*********************/
float32_t       Motor1_Over_CurrentA = 0.6;         //the maximum soft over current protection
uint32_t        Motor1_Max_Dc_VoltageV = 420;        //the maximum value of DC voltage is 400V
uint32_t        Motor1_Min_Dc_VoltageV = 20;         //the minimum value of DC voltage is 140V
uint32_t        Motor1_Error_Keep_TimeS = 10;       // the maximum time error maintain
```

7.1.4.5 Configure Running Parameters

1. Set the field weaken current:

    a. If the voltage of power supply and back electromotive force can run the motor at maximum speed, set the field weaken current to zero;

    b. Otherwise, set the field weaken current according to your requirements. Theoretically, the field weaken current is not over 60% of total current.

2. Debug and confirm the maximum of dead time compensation and threshold according to motor characteristic:

    a. Dead time compensation should depend on IPM or MOSFET hardware performance. Theoretically, the time should be as short as possible in order to eliminate harmonic and control error;

3. Set the motor maximum current according to system power;

4. Configure motor start-up frequency and current loop carrier frequency according to motor characteristic:

    a. Please use the default parameters if the electromagnetic noise and start-up performance are acceptable;

    b. If you enabled the function of dynamic adjustment the carrier frequency online, this function can turn the motor run with proper carrier frequency to eliminate electromagnetic.

5. Modify the speed range according to motor characteristic;

6. All the parameters in Figure 7-11support change online, you can modify it to control your motor to get the best state conveniently.

Figure 7-11: Motor Running Parameters

```
/*************motor1 running parameters***************/
float32_t       Motor1_Field_Value = 0;             //the maximum current of field value
float32_t       Motor1_Comp_MaxA   = 0.2;          //Dead time compensation offset banlance current.
float32_t       Motor1_Is_MaxA    =  0.7;          //the motor running maximum torque current
float32_t       Motor1_Dead_TimeUS =2.0;            //the dead time value
uint32_t        Motor1_PWM_Carry_Frequency =15000;  //the motor start running PWM carry frequency
uint32_t        Motor1_Current_Carry_Frequency=5000;//the current implement frequency
uint32_t        Motor1_Speed_Max_RPM    = 1000;     //the motor running maximum speed of RPM
uint32_t        Motor1_Speed_Min_RPM    = 180;      //the motor running minimum speed of RPM
uint32_t        Motor1_Rotor_Direction = 1;         //the motor running derection 0:CW 1:CCW
uint32_t        Motor1_Running_Level   =  4;        //the motor running stage
uint32_t        Motor1_Dynamic_PWM_Enable =0;       //the motor carry frequency switch 0:disable 1:enable
uint32_t        Motor1_PWM_Table[30] = {            //Motor1 PWM Carry Frequency/1000
```

7.1.4.6 Hardware Parameters Configuration

For the detail of hardware parameter configuration, definition and calculation, please refer to the chapter 5.2.

7.2 Motor Start-up

7.2.1 Code Rebuild

In IAR environment, select the project name “ArmInverterPlantform”, and right-click. Then select “Rebuild All”, or click “Project->Rebuild All”, as shown in Figure 7-12 and Figure 7-13.

Figure 7-12: Code Rebuild (1)

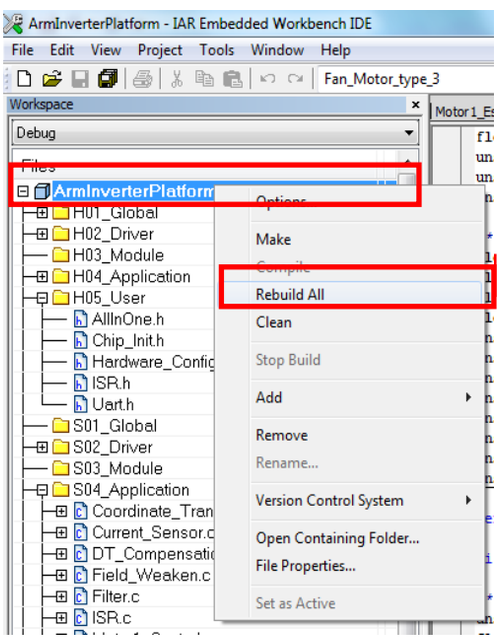
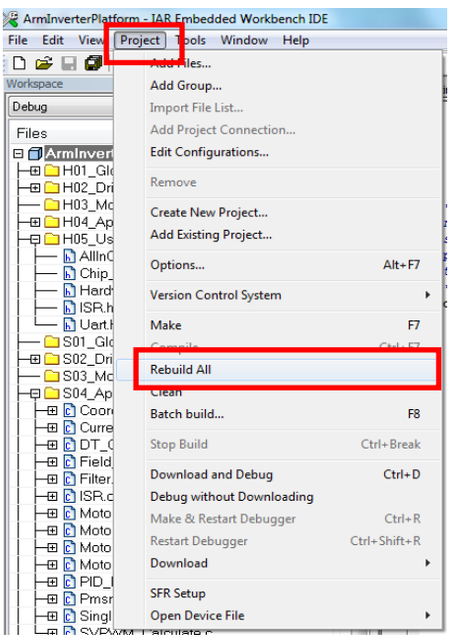


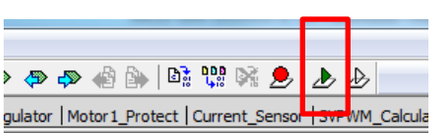
Figure 7-13: Code Rebuild (2)



7.2.2 Download Code

When rebuild code passed, left click “Project->Download and Debug” or according to Figure 7-14, click the shortcut button to enter the debug interface.

Figure 7-14: Download Code

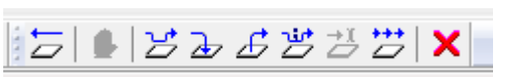



```
*******************************
```

7.2.3 Program and Motor Running

As shown in Figure 7-15 click “  ” or press “F5” to run program, in this status, program can respond to your command and return the status and data.

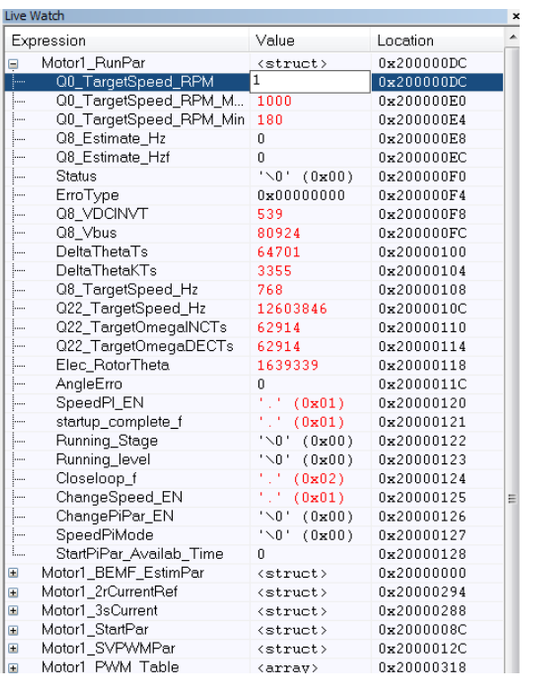
Figure 7-15: Running Button



Right click “View->Live Watch” to enter the interface, type the global variable names in “Live Watch” window.

Such as: “Motor1_RunPar”, as shown in Figure 7-16. Expansion the structure variable “Motor1_RunPar”, type a non-zero value to “Q0_TargetSpeed_RPM” and press “Enter”, the motor can be start-up. When motor start-up finished, we can modify the parameters as mentioned before according to the current waveform in oscilloscope.

Figure 7-16: Variable Watch Window



7.2.4 Debug Motor Running Status

7.2.4.1 Debug Start-up

Set different parameters according to start-up modes, for example:

1. Prepare Close Loop Start-up Mode (default mode)

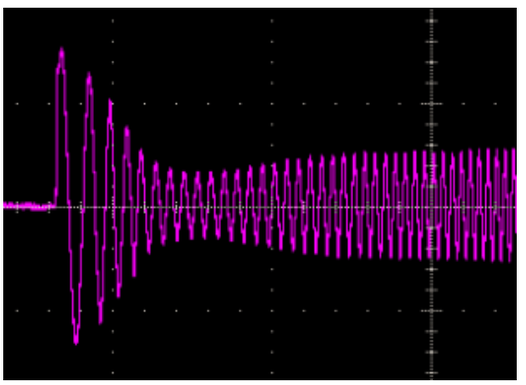
Set the parameters as shown in Figure 7-17:

Figure 7-17: Parameters Watch Window

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_OpenLoop_Inc_Current...	1.00000001E-1
Motor1_OpenLoop_CurrentA	3.00000012E-1
Motor1_Orient_TimeS	0.0
Motor1_Force_Inc_Speed_Hz	1.00000001E-1
Motor1_CloseLoop_Speed_Hz	2.0
Motor1_Initial_Speed_Pre_Clos...	2.0
Motor1_Detect_Direction_TimeS	0.5
Motor1_Pre_CloseLoop_Curren...	2.00000003E-1

Figure 7-18: Start-up Current Waveform (100mA/1s/scale)



2. Start-up with no orient and open loop mode

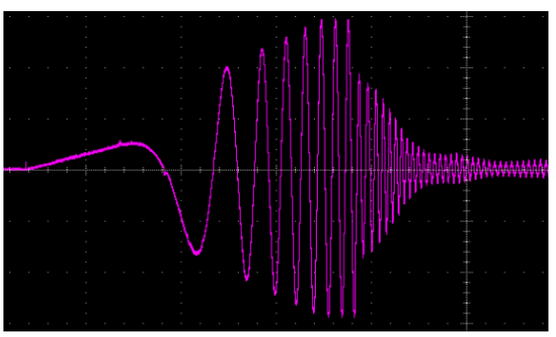
Set the parameters as shown in Figure 7-19. (Modify the “Motor1_Orient_TimeS” & “Motor1_Initial_Speed_Pre_Close_Hz” value on the basis of Figure 7-17)

Figure 7-19: Start-up with No Orient and Open Loop Mode Parameters

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_OpenLoop_Inc_Current...	1.00000001E-1
Motor1_OpenLoop_CurrentA	3.00000012E-1
Motor1_Orient_TimeS	1.0
Motor1_Force_Inc_Speed_Hz	1.00000001E-1
Motor1_CloseLoop_Speed_Hz	2.0
Motor1_Initial_Speed_Pre_Clos...	1.0
Motor1_Detect_Direction_TimeS	0.5
Motor1_Pre_CloseLoop_Curren...	2.00000003E-1

Figure 7-20: Start-up with No Orient and Open Loop Current waveform (100mA/1s/scale)



3. Start-up with orient and open loop mode

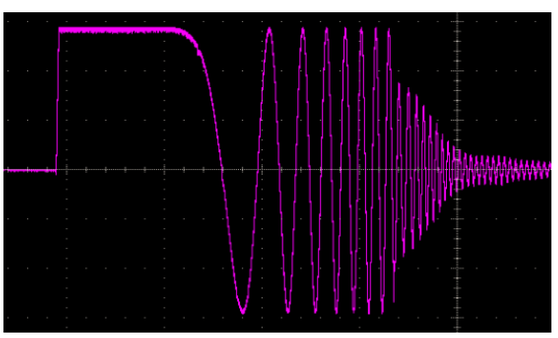
Set the parameters as shown in Figure 7-21. (Modify the “Motor1_OpenLoop_Inc_Current_APS” value on the basis of Figure 7-19)

Figure 7-21: Start-up with Orient and Open Loop Mode Parameters

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_OpenLoop_Inc_Current...	10.0
Motor1_OpenLoop_CurrentA	3.00000012E-1
Motor1_Orient_TimeS	1.0
Motor1_Force_Inc_Speed_Hz	1.00000001E-1
Motor1_CloseLoop_Speed_Hz	2.0
Motor1_Initial_Speed_Pre_Clos...	1.0
Motor1_Detect_Direction_TimeS	0.5
Motor1_Pre_CloseLoop_Curren...	2.00000003E-1

Figure 7-22: Start-up with Orient and Open Loop Current Waveform (100mA/1s/scale)



7.2.4.2 Debug PI Parameters

Adjust the PI controller parameters according to motor status and current waveform, for example:

1. Speed Loop

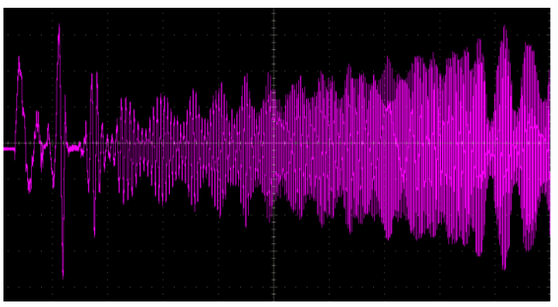
a. Set a group over large PI parameters, as shown in Figure 7-23, the motor current waveform as shown in Figure 7-24.

Figure 7-23: Speed Loop PI Parameters (Over large)

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_Skp	1.0
Motor1_Ski	4.99999988E-3

Figure 7-24: Motor Current Waveform in PI Parameters Over Large (100mA/1s/scale)



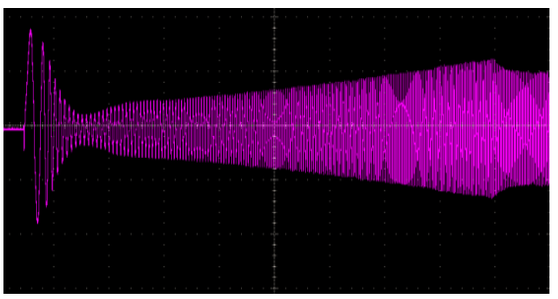
b. Set a group PI parameters appropriately, as shown in Figure 7-25; the motor current waveform is shown in Figure 7-26.

Figure 7-25: Speed Loop PI Parameters

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_Skp	1.00000001E-1
Motor1_Ski	5.00000023E-4

Figure 7-26: Motor Current when PI Parameters appropriately (100mA/1s/scale)



2. Current Loop

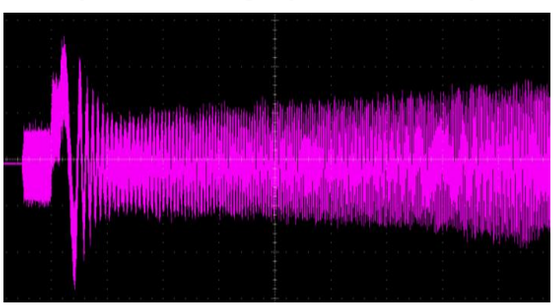
a. Set a group over large D&Q Axis PI parameters, as shown in Figure 7-27, the motor current waveform is shown in Figure 7-28, the current ripple and electromagnetic noise is very heavy.

Figure 7-27: D&Q Axis PI Parameters (PI parameters over large)

Live Watch

Expression	Value
⊞ Motor1_RunPar	<struct>
Motor1_Dkp	2000.0
Motor1_Dki	42.0
Motor1_Qkp	2000.0
Motor1_Qki	42.0

Figure 7-28: Motor Current when D&Q Axis PI parameters larger (100mA/1s/scale)

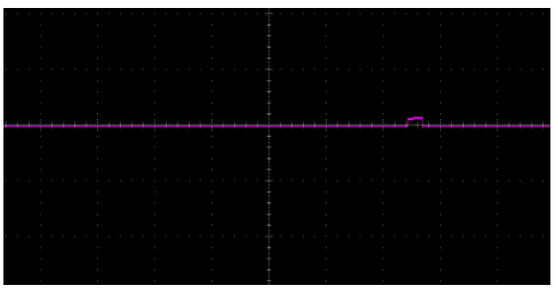


b. Set a group too small D&Q Axis PI parameters, as shown in Figure 7-29, the motor current can't follow the reference current, the system enter error, the motor current waveform is shown in Figure 7-30.

Figure 7-29: D&Q Axis PI Parameters (PI parameters too small)

Motor1_Dkp	1.00000001E-1
Motor1_Dki	2.19999998E-3
Motor1_Qkp	1.00000001E-1
Motor1_Qki	2.19999998E-3
Motor1_Pre_CloseLoop_Curren...	6.99999988E-1
Motor1_2rCurrentRef	<struct>
Q12_d	0
Q12_q	998

Figure 7-30: Motor Current when D&Q Axis PI parameters Smaller (100mA/1s/scale)



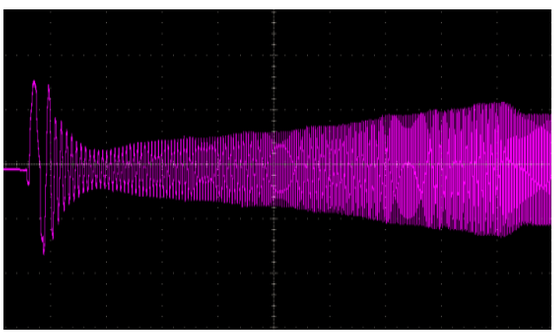
c. Set a group appropriately D&Q Axis PI parameters, as shown in Figure 7-31, the motor current ripple is very little and follow the reference current very nice, the motor current waveform is shown in Figure 7-32.

Figure 7-31: D&Q Axis PI Parameters (PI parameters appropriately)

Live Watch

Expression	Value
Motor1_RunPar	<struct>
Motor1_Dkp	300.0
Motor1_Dki	6.0
Motor1_Qkp	300.0
Motor1_Qki	6.0
Motor1_Pre_CloseLoop_Curren...	2.00000003E-1

Figure 7-32: Motor Current when D&Q Axis PI parameters appropriately (100mA/1s/scale)



7.2.5 VSP Modify Motor Speed

To achieve the VSP modify motor speed, you need to follow below steps:

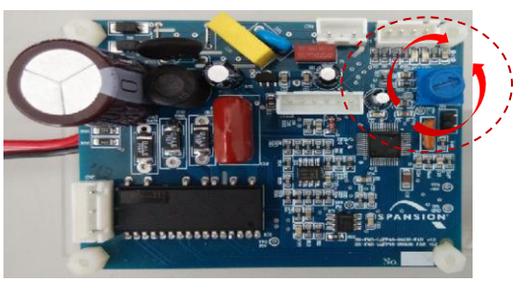
Step 1: Define the parameters "SPEED_FROM_OUT" and "VSP_MOTOR_START_V" in 'Hardware_Congig.h' file, as shown in below:

```
#define SPEED_FROM_OUT      1
#define VSP_MOTOR_START_V   0.8
```

Step 2: "Rebuild All" the project and "Download and Debug", close the IAR workbench;

Step 3: Twist the resistance knob clockwise or anti-clockwise slowly, as shown in Figure 7-33 to record the motor speed;

Figure 7-33: Knob of Demo Board



7.2.6 Torque Control Mode

To achieve torque control mode, you need to follow below steps:

Step1: Run motor in "speed" mode at any speed;

Step2: set the value of "Motor1_Startup.PMSM_Control_mode" to "1" in "Live Watch" window;

Step3: change the value of "Motor1_2rCurrentRef.Q12_q"; in this step, motor speed is determined by the value of "Motor1_2rCurrentRef.Q12_q";

Note: the "Motor1_2rCurrentRef.Q12_q" value need to be greater than 500 and less than 1400;

8. System Error Code Definition

Table 8-1: Error Code Table

Error Code	Error Type	Remark
0x00	Motor initial state or status normal	none
0x01	Over DC voltage	none
0x02	Under DC voltage	none
0x04	Soft over current	none
0x08	Hardware over current	none
0x10	Motor Loss phase	none
0x20	No connect the motor	none
0x40	AD offset error	none
0x80	Software watch dog	none
0x100	Lock the motor	none
0x200	Handle the error	none
0x400	Hardware watchdog	none

9. Additional Information

For more Information on Spansion semiconductor products, visit the following websites:
English version address:
http://www.spansion.com/Products/microcontrollers/

Chinese version address:
http://www.spansion.com/CN/Products/microcontrollers/

Please contact your local support team for any technical question
America: Spansion.Solutions@Spansion.com
China: mcu-ticket-cn@spansion.com
Europe: mcu-ticket-de@spansion.com
Japan: mcu-ticket-jp@spansion.com
Other: http://www.spansion.com/Support/SES/Pages/Ask-Spansion.aspx

S6E1A1_AN710-00004-1v0-E

Spansion • Application Note

FM3/0+ Family
32-BIT MICROCONTROLLER
Fan Motor User Manual

February 2015 Rev. 1.0

Published: Spansion Inc.
Edited: Embd System Plat Dev-Embd Solution

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.